іністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА
ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Об’єктно-орієнтоване програмування”

ДОВІДНИК ФІЛАТЕЛІСТА

Керівник Ляпота В. М.

Студент гр. ПІ-15-2 Володін Д.О.

Комісія:

Зав. каф. ПІ, Дудар З.В.

к. т. н, проф., Бондарєв В.М.

асистент Ляпота В.М.

Харків 2016

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра: ***Програмної інженерії***

Дисципліна: ***Обєктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 . Група *ПІ-15-2* . Семестр 2 .

ЗАВДАННЯ

на курсовий проект студента

Володіна Дмитра Олександровича

(Прізвище, Ім'я, По батькові)

1. Тема проекту: ***Довідник філателіста.*** *Марки: страна, нарицательная стоимость, год выпуска, тираж, особенности. Филателисты: страна, имя, контактные координаты, наличие редких марок в коллекции. Собственная коллекция.*
2. Термін здачі студентом закінченого проекту: ***"28 " - червня - 2016 р.***
3. Вихідні дані до проекту: *Специфікація програми, методичні вказівки до виконання курсової роботи*
4. Зміст розрахунково-пояснювальної записки: *Вступ, специфікація програми, проектна специфікація, інструкція , висновки.*

КАЛЕНДАРНИЙ ПЛАН

№	*Назва етапу*	*Термін виконання*
1	Видача теми, узгодження і затвердження теми	1-03-2016 р.
2	Аналіз предметної області	1-03-2016 – 15-03-2016 р.
3	Розробка постановки задачі	20-03-2016 – 2-04-2016 р.
4	Розробка об’єктної моделі	5-04-2016 – 11-04-2016 р.
5	Кодування програмної системи	20-04-2016 – 5-05-2016 р.
6	Тестування і доопрацювання розробленої програмної системи.	9-05-2016 – 27-05-2016 р.
7	Оформлення пояснювальної записки	25-05-2016 – 27-05-2016 р.
8	Публічний захист проекту перед комісією	28-05-2016 р.

Студент: Володін Д. О.

Керівник: Ляпота В. М.

«28 » лютого 2016р.

РЕФЕРАТ

Пояснювальна записка: 26 с., 17 рис., 1 додаток.

Мета роботи: закріплення знань, отриманих під час вивчення дисципліни «Об’єктно-орієнтовне програмування», шляхом розробки програмної системи під назвою «Довідник філателіста».

Методи розробки: Microsoft VisualStudio 2015, Windows Forms, .NET Framework 4.5

В результаті розробки отримана програмна система під назвою «Довідник філателіста» для роботи з інформаційною базою даних, чию роль виконують *dat файли. Розроблена програма дає швидкий та зручний доступ до коллекцій марок, інформації про коллекціонерів та марки, які існують в базі. Також є можливість додавання нових коллекціонерів, марок, коллекцій; їх редагування або видалення, якщо існує така необхідність. Экспортування данних у формат TXT.

КУРСОВА РОБОТА, ДОВІДНИК ФІЛАТЕЛІСТА, КОЛЛЕКЦІОНЕРИ, МАРКИ, ДОВІДНИК, ПРОГРАММА.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Филателисты - собиратель почтовых марок и других знаков почтовой оплаты, руководствующийся при создании своей коллекции принципами филателии. При этом филателист не просто коллекционирует марки как таковые, а исследует в рамках выбранного филателистического направления совокупность знаков почтовой оплаты и других филателистических материалов, изучает историю и развитие почты, оформляет и компонует свою коллекцию для участия в филателистических выставках. Слово «филателист» является производным от «филателии» и обозначает «любящий знаки почтовой оплаты».

Собирание почтовых марок как вид коллекционирования ведёт свой отсчёт с момента выпуска в 1840 годупервых в мире марок. По одной из версий, первым человеком, систематически коллекционировавшим марки, был парижский гравёр Мансен. В дальнейшем филателия превратилась в один из самых популярных видов коллекционирования, а филателисты стали объединяться в национальные и международные филателистические организации. В мире насчитываются миллионы зарегистрированных филателистов. Собирателей марок, не числящихся ни в каких обществах, во много раз больше. Интерес к коллекционированию почтовых марок проявляли русские литераторы А. Чехов, А. Блок и М. Горький, академики И. П. Бардин и И. П. Павлов, командир крейсера «Варяг» В. Ф. Руднев, президент США Ф. Рузвельт, певец Э. Карузо и многие другие.

В данной курсовой работе требуется разработать программу «Справочник филателиста». Целью данной курсовой работы является разработка программы-справочника, с помощью которого можно создавать и просматривать марки,

информацию о коллекционерах и их коллекциях, осуществлять поиск по имеющейся базе данных. За основу взят объектно-ориентированный подход.

Задачи выполнения работы:

a) исследование предметной области с целью выявления основных принципов данной сферы;

b) проектирование иерархии классов, интерфейсов, взаимодействия компонентов на основе выделенных принципов и данных средств;

c) использование встроенные элементы среды для структуризации классов и оптимизации кода;

d) применение принципа инкапсуляции к классам;

e) реализовать программное взаимодействие с базой данной формата *txt.

Объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. В данном подходе основными концепциями являются понятия объектов и классов.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования. Использование этого подхода предоставляет программисту широкие возможности в функциональности и в сопровождаемости проекта.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015 Proffesional. Visual Studio представляет собой интегрированную среду разработки программ, созданную корпорацией Microsoft. Язык программирования C#.

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ

1.1 Главное окно, добавление коллекционера, марок

Работа с программой начинается с главного окна (рис. 1.1). Здесь пользователь может увидеть список всех коллекционеров; просмотреть коллекцию каждого коллекционера и увидеть информацию про каждую выбранную марку; или же перейти по нужной вкладке меню. Интерфейс программы прост в использование и интуитивно понятен. Это является необходимым условием при разработке пользовательских интерфейсов [3].

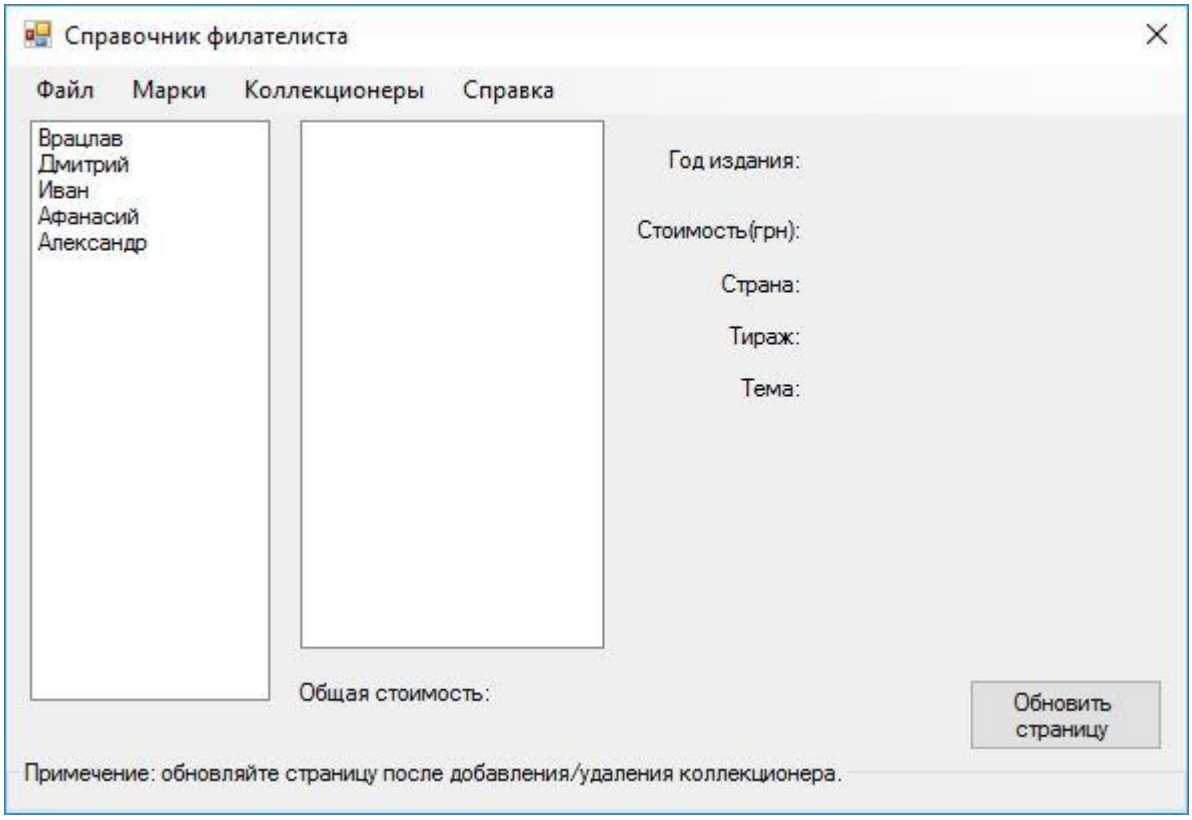

Рисунок 1.1 – Главное окно

Для добавления нового коллекционера или марки пользователю необходимо открыть вкладку меню «Файл», далее «Добавить…» и выбрать нужный элемент (рис 1.2).

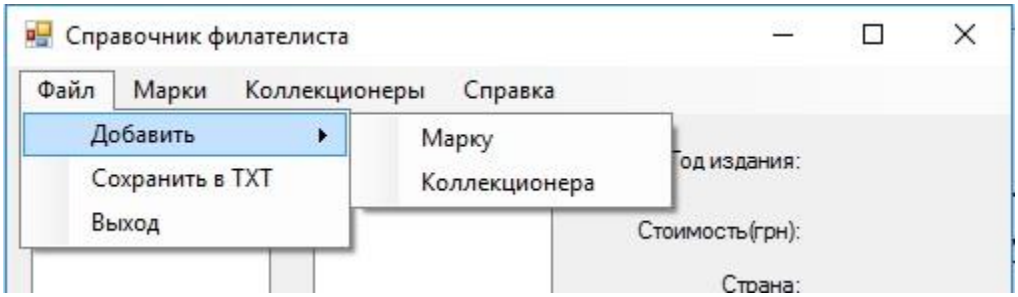

Рисунок 1.2 – Добавление нового элемента

После появляется окно добавления коллекционера (рис. 1.3) или марки (рис. 1.4). При верном заполнение всех полей и нажатие на кнопку «Добавить» новый элемент будет добавлен.

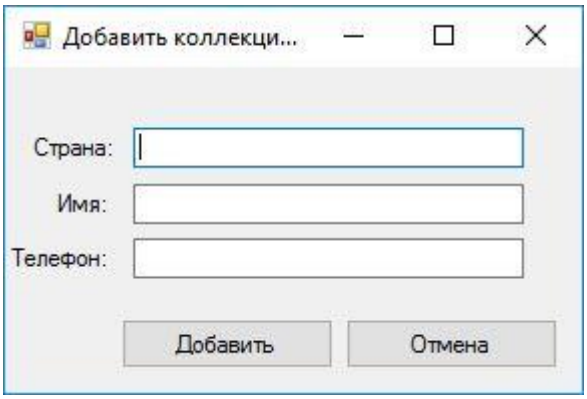

Рисунок 1.3 – Добавление коллекционера

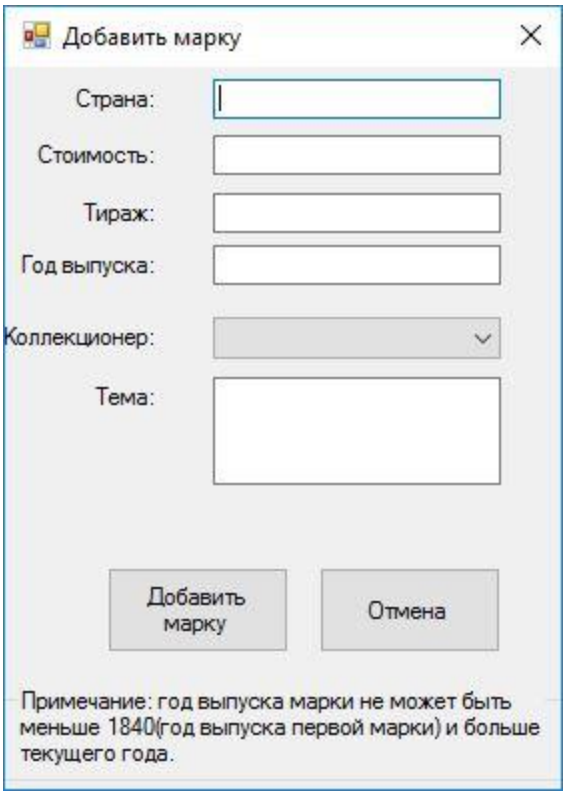

Рисунок 1.4 – Добавление марки

1.2 Коллекционеры, марки

При выборе вкладки меню «Коллекционеры» пользователю будет представлено окно для работы с имеющимися авторами (рис 1.5). В табличном виде размещен список коллекционеров и информация о них. Над таблицей находится блок поиска, в котором осуществляется пиоск по различным критериям: имя, страна, телефон.

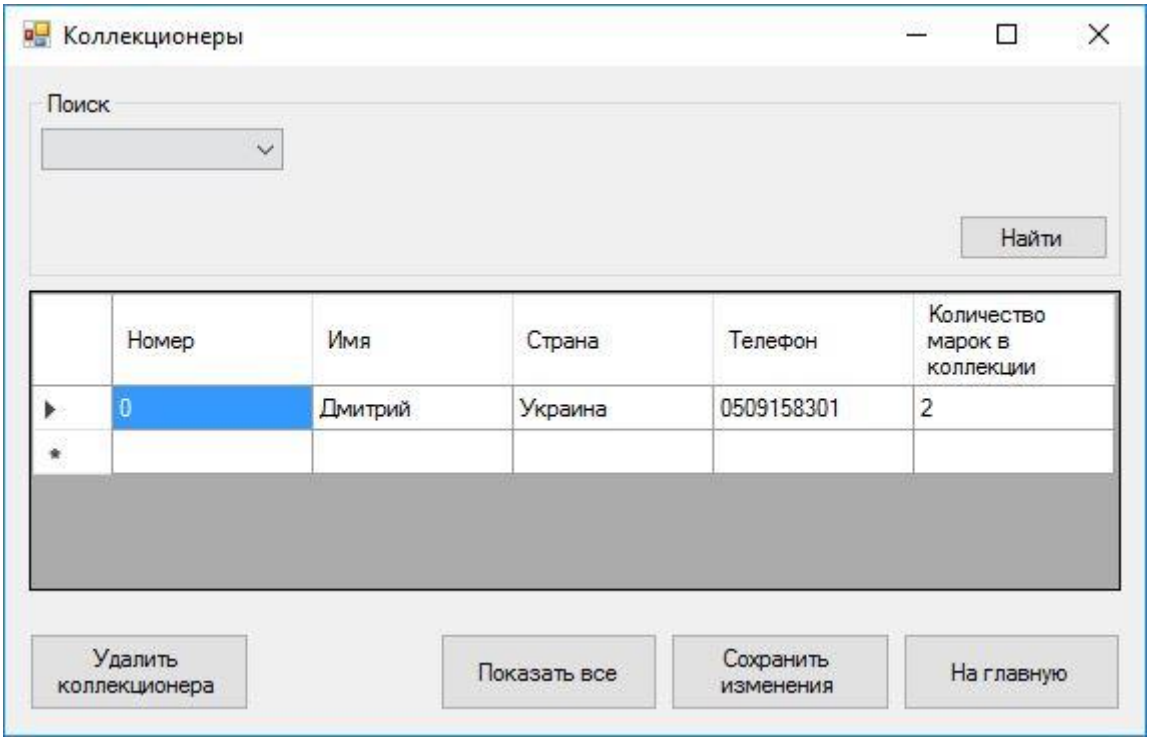

Рисунок 1.5 – Окно «Коллекционеры»

Также в этом окне есть возможность удалять и изменять информацию. При нажатии на графу «Телефон» появляется возможность его редактирования, если все введено верно, то при нажатии кнопки «Сохранить изменения» измененные данные сохранятся.

Окно с марками (рис. 1.6), которое появляется при выборе вкладки меню «Марки», во многом схоже с окном «Коллекционеры». Здесь также в таблице размещен список всех марок, есть кнопка удаления, поиск по заданным критериям.

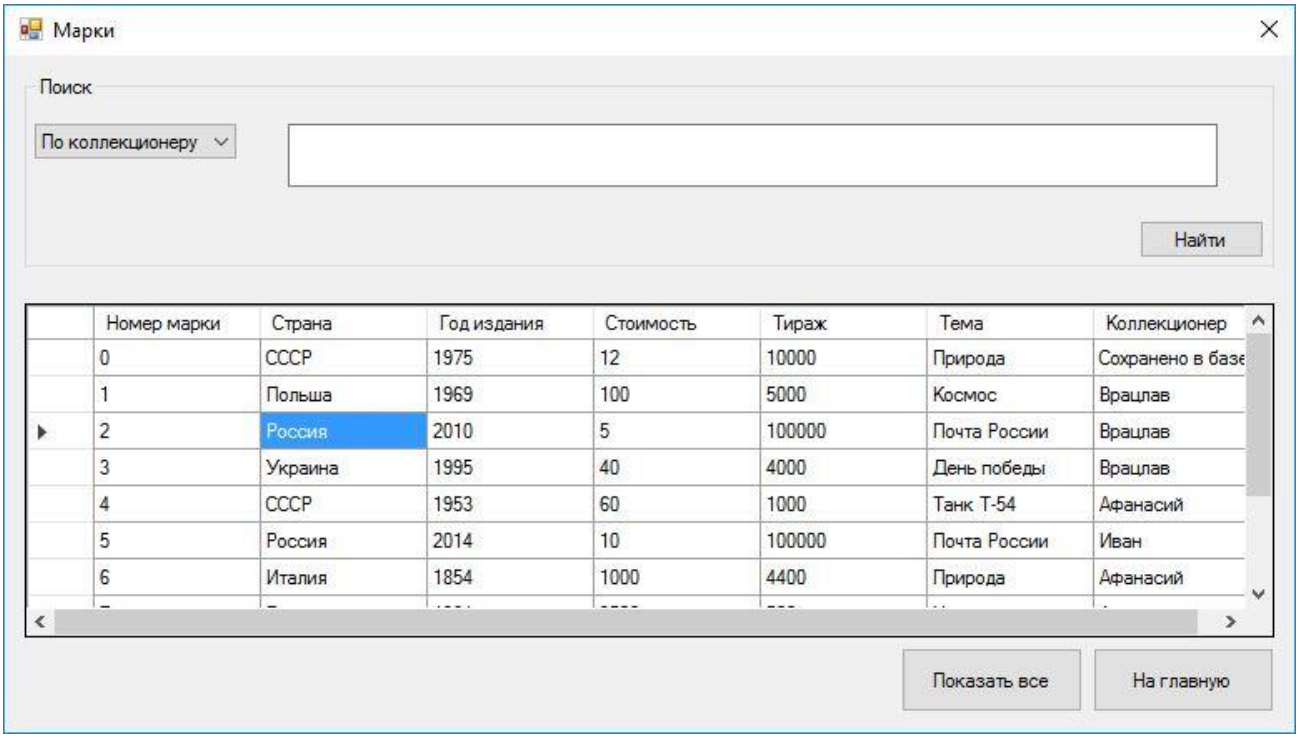

Рисунок 1.6 – Окно «Марка»

1.3 Работа с коллекциями, справка

Нажав кнопкой мыши в блоке с именами коллекционеров в главном меню на одного из коллекционеров, во втором блоке появляется коллекция марок выбранного коллекционера (рис. 1.7), в которой можно, нажав на марку, просмотреть информацию или удалить её из коллекции. Удалив её здесь, данные о марке останутся в базе данных, но исчезнут из коллекции выбранного коллекционера.

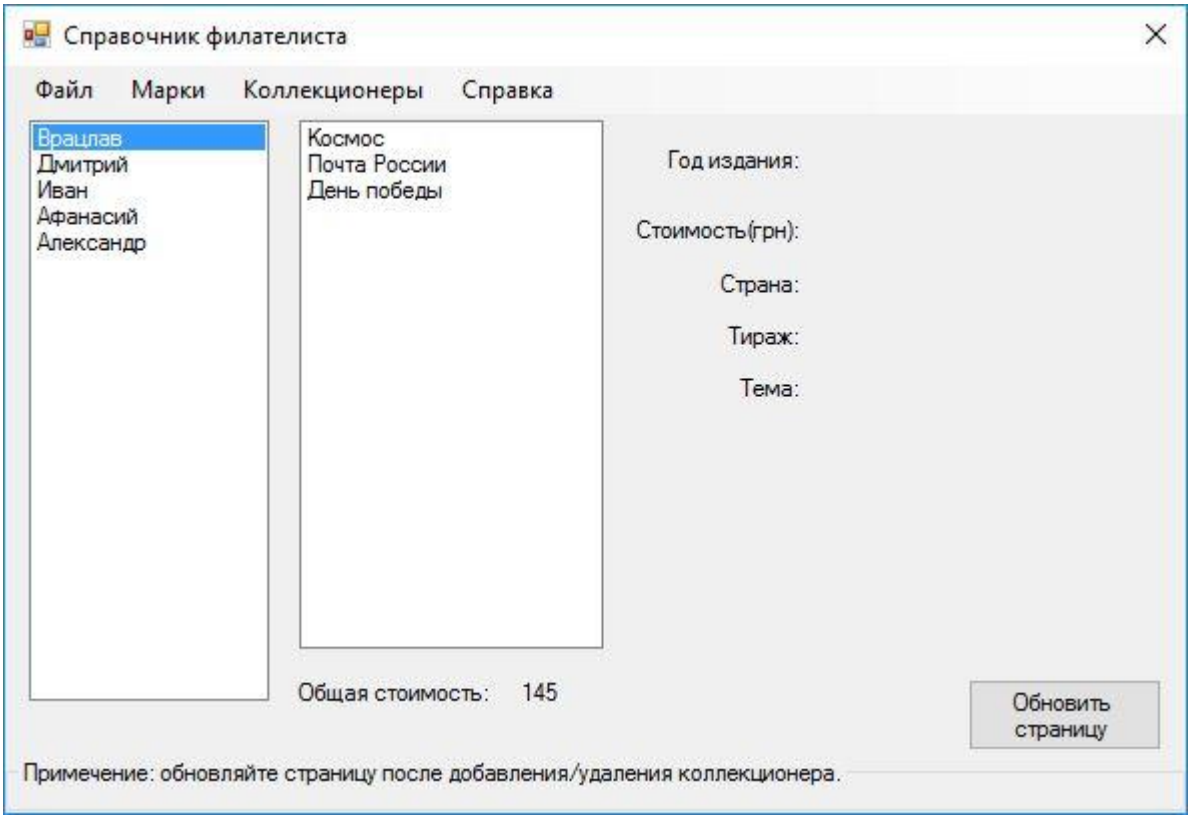

Рисунок 1.7 – Коллекция выбранного коллекционера

При нажатие левой кнопкой мыши на марку в блоке 2 появляется информация о ней (рис. 1.8).

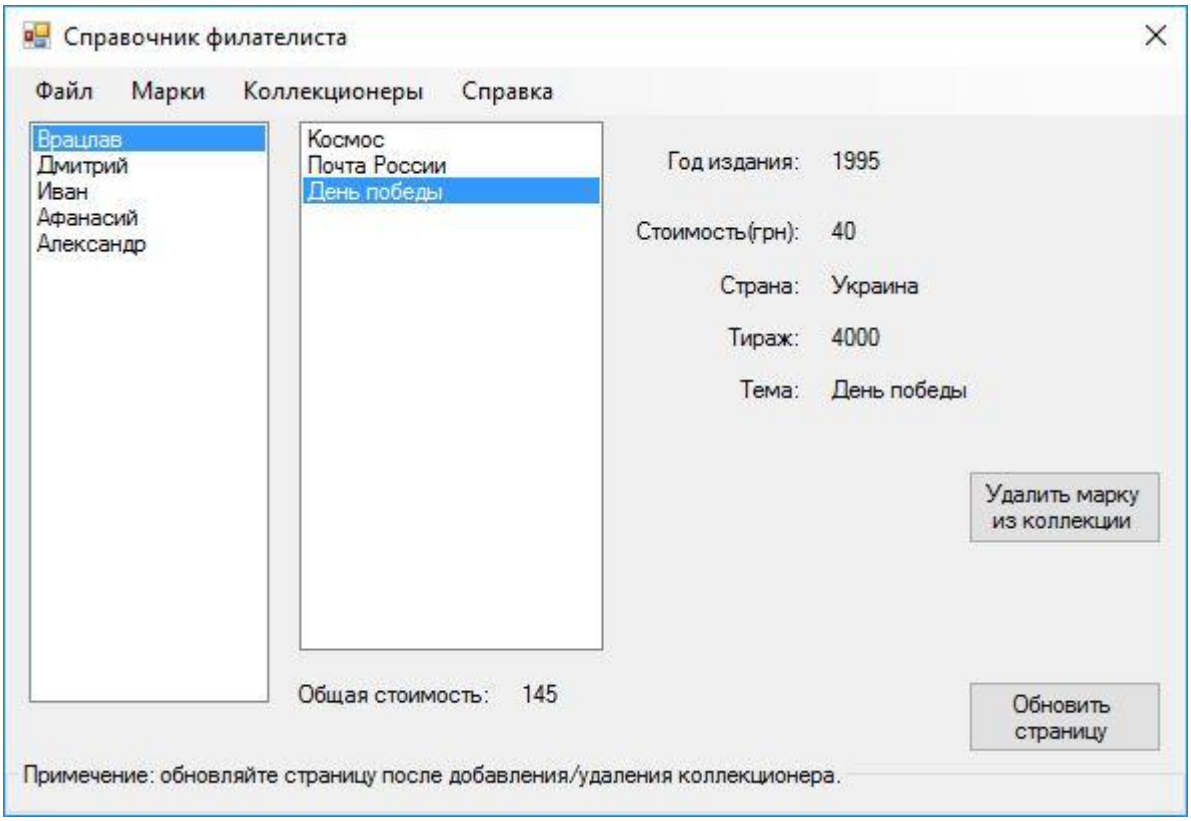

Рисунок 1.8 – Выделенная марка

Инструкция пользователя размещена во вкладке «Справка», подвкладка «Помощь» (рис. 1.9). Информацию о программе (рис. 1.10) можно узнать, выбрав «О программе…» в той же вкладке.

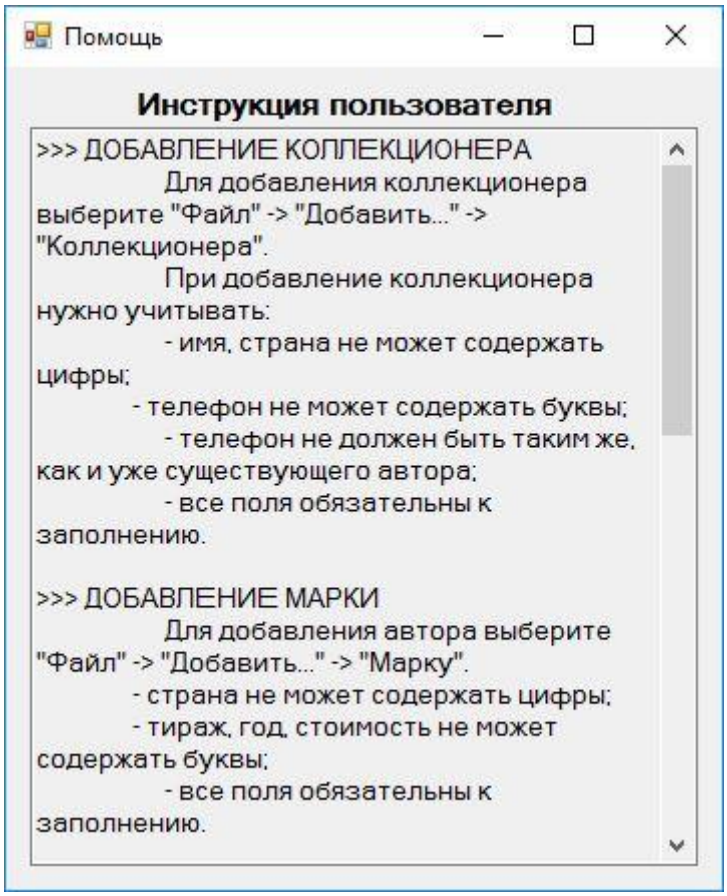

Рисунок 1.9 – Инструкция пользователя

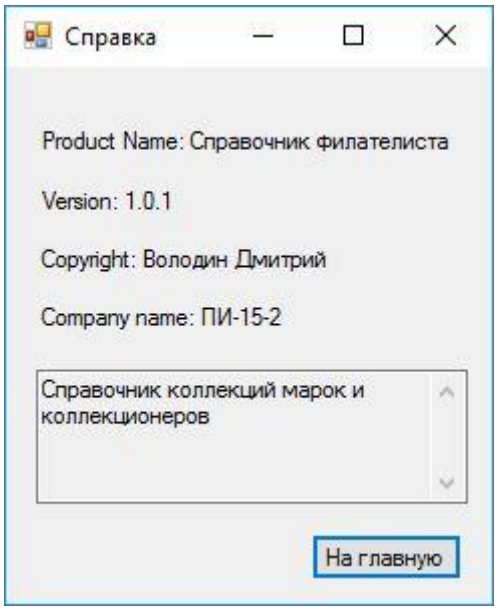

Рисунок 1.10 – О программе

2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ

2.1 Объектная модель программы

Основным конструктивным элементом в языках ООП является модуль (module), представляющий собой логически связанную совокупность классов и объектов, а не подпрограмм, как в более ранних языках. В графическом интерфейсе данной программы содержатся несколько форм заполнения и редактирования данных, следовательно в проекте присутствуют специальные вспомогательные функции, которые считывают данные, заполняют коллекции, сохраняют данные. Считывание данных происходит при загрузке окон. Методом записи и считывания соответственно является сериализация и десериализация.

При записи данных в документ с расширением *dat сериализируются соответственные классы ListOfCollectors, ListOfMarks (рис. 2.1). При запуске окон проводится их десериализаци. В этих классах реализованы методы поиска, создания, удаления, сериализации, десериализации объектов. Классы наследуются от класса List, в котором реализованы некоторые общие методы для каждого из классов.

Рисунок 2.1 – Классы для работы с данными

Наследование позволяет определять в родительском классе общую функциональность, которая может применяться и, возможно, изменяться в дочерних классах.

Коллекции состоят из объектов следующих классов: Collector, Marka (рис. 2.2).

Рисунок 2.2 – Классы данных

В каждом из этих классов содержаться свойства, в которых будет храниться информация о объекте и конструктор (рис. 2.3). В классе Marka перекрыт метод ToString.

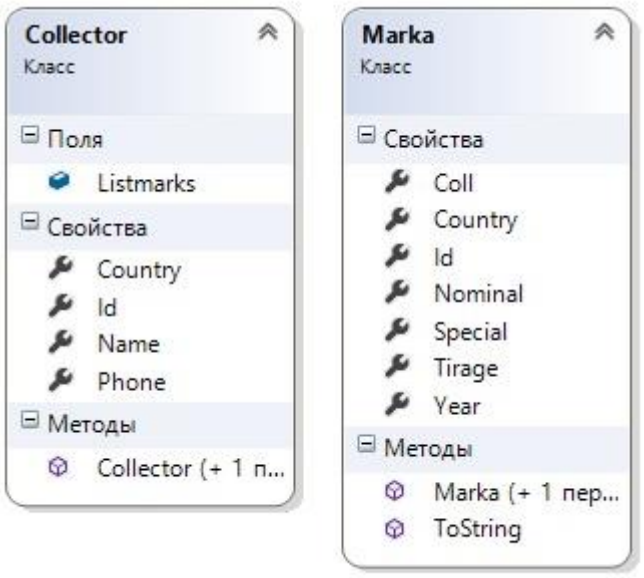

Рисунок 2.3 – Классы Collector, Marka

Класс ListOfCollectors предназначен для работы с информацией об коллекционерах. Он наследует класс List в котором есть почти все необходимые для программы функции, поэтому реализация полей и свойств в нем не

обязательна. В нем реализован метод IsCopy, который проверят вхождение одинаковых авторов в ListOFCollectors (рис. 2.4).

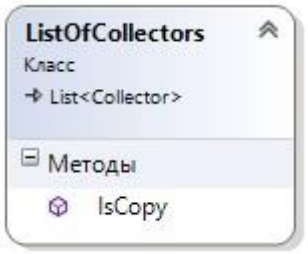

Рисунок 2.4 – Методы класса ListOfCollectors

Класс ListOfMark (рис. 2.5) предназначен для работы с информацией про марки. Аналогично с ListOfCollectors, наследуется от класса List и не требует никаких дополнительных полей, свойст и методов.

Рисунок 2.5 – Класс ListOfMarks

Класс Serial (рис. 2.6) предназначен для сериализации и десериализации данных. Содержит два поля классов ListOfMarks и ListOfCollectors. Также содержит методы самой сериализации и десериализации для обоих классов, представленных выше. (рис. 2.7).

Рисунок 2.6 – Поля класса Serial

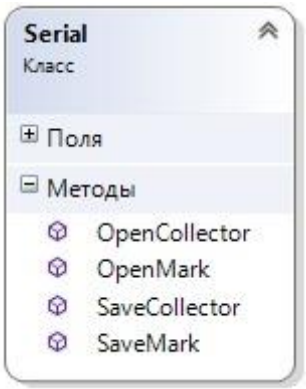

Рисунок 2.7 – Методы класса Serial

2.2 Реализация функций программы

Как упоминалось ранее, данная программа состоит из нескольких компонентов, которые тесно взаимосвязаны друг с другом. В этом разделе будут рассмотрены большинство из этих функций. Полный набор методов можно увидеть в коде программы (приложение А).

Одним из важнейших методов программы являются методы сериализации и десериализации объектов. Рассмотрим их на примере одного класса (рис. 2.8, рис. 2.9).

```
public static void SaveCollector()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Collectors.dat",
FileMode.OpenOrCreate))
                {
                    binFormat.Serialize(fStream, collectorsList);
                }
            }
            catch { }
        }
```

Рисунок 2.8 – Сериализация

```
public static void OpenCollector()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Collectors.dat",
FileMode.Open))
                {
                    collectorsList =
(ListOfCollectors)binFormat.Deserialize(fStream);
                }
            }
            catch { }
         }
```

Рисунок 2.9 – Десериализация

Также очень важным является метод поиска (рис. 2.10), который позволяет искать марки по стране издания, коллекционеру, теме и является не чувствительным к регистру. Метод находится в форме AllMark.

```
private ListOfMarks SearchText()
        {
            ListOfMarks searchList = new ListOfMarks();
            if (textbox.Text.Length == 0)
            {
                MessageBox.Show("Вы ничего не ввели");
            }
            else
            {
                foreach (Marka mark in Serial.marksList)
                {
                    if ((n == 4 &&
mark.Coll.Name.ToLower().Contains(textbox.Text.ToLower())) || (n == 5 &&
mark.Special.ToLower().Contains(textbox.Text.ToLower())) || (n == 6 &&
mark.Country.ToLower().Contains(textbox.Text.ToLower())))
                    {
                        searchList.Add(mark);
                    }
                    else
                    {
                        continue;
                    }
                }
            }
            return searchList;
        }
```

Рисунок 2.10 – Метод поиска

Метод FillGridView (рис. 2.11) необходим для формирования таблицы марок с данными о них. Метод запускается при открытии формы AllMarks и берет данные из десериализованной коллекции marksList.

```
        public void FillGridView()
        {
            MarksGridView.Rows.Clear();
            Serial.OpenMark();
            foreach (Marka marka in Serial.marksList)
            {
                MarksGridView.Rows.Add(marka.Id, marka.Country, marka.Year,
marka.Nominal, marka.Tirage, marka.Special, marka.Coll.Name);
            }
        }
```

Рисунок 2.11 – Метод FillGridView

Метод записи базы марок в *txt файл (рис. 2.12) вызывается при нажатии на подменю «Сохранить в TXT» и записывает в файл ListOfMarks на диск D. Запись происходит благодаря перекрытию метода ToString() (рис. 2.13) в классе Marka.

```
  private void сохранитьВTXTToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (StreamWriter writer = new StreamWriter("d:\\ListOfMarks.txt",
false, Encoding.UTF8))
            {
                foreach (Marka mark in Serial.marksList)
                {
                    writer.WriteLine(mark.ToString());
                }
            }
            MessageBox.Show("Сохранение прошло успешно.");
         }
```

Рисунок 2.12 – Метод сохранитьВTXT

```
       public override string ToString()
        {
            return String.Format(" Страна: {0}, Номинальная стоимость: {1}, Год:
{2}, Тираж: {3}, Особенность: {4} ", Country, Nominal, Year, Tirage, Special);
        }
```

Рисунок 2.13 – Перекрытие метода ToString

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для того, чтоб добавить коллекционера нужно выбрать меню "Файл", далее "Добавить...", после "Коллекционера". При добавлении коллекционера нужно учитывать:

a) Имя не может содержать цифры;

b) Страна также не может содержать цифры;

c) Телефон не должен быть таким же, как и уже существующего автора;

d) Телефон не должен содержать буквы;

e) Все поля должны быть заполнены;

Для добавления марки выберите кнопку меню "Файл", после "Добавить...", далее "Марку". Будет загружен список коллекционеров, выбрать нужного в выпадающем меню. Далее заполняем поля.

Год создания не может быть больше текущего года;

a) Страна должна содержать только буквы;

b) Тираж, стоимость, год издания должны содержать только цифры;

c) Все поляобязательны к заполнению.

В меню "Коллекционеры" предоставлен список всех коллекционеров, а также информация о них. Здесь пользователь может изменить/удалить информацию про коллекционера (кнопка "Удалить коллекционера", изменение номера телефона в графе телефон). Удаление коллекционера влечет за собой удаление его марок из базы данных. В окне расположен динамический поиск. Он становиться доступным только после выбора критерия поиска. Для показа всех существующих коллекционеров нужно нажать кнопку "Показать все", которая расположена в нижнем левом углу формы.

В меню "Марки" предоставлен список всех марок, а также информация о них. Здесь пользователь может удалить марку (кнопка "Удалить марку"). Удаление марки влечет за собой удаление её из базы данных марок, но она останется в частных коллекциях коллекционеров. В окне расположен динамический поиск. Он становиться доступным только после выбора критерия поиска. Для показа всех существующих марок нужно нажать кнопку "Показать все", которая расположена в нижнем левом углу формы.

ВЫВОДЫ

Результатом выполнения данной курсовой работы стала программа – справочник «Справочник филателиста». Данная программа предназначена для удобного поиска марок и их коллекционеров.

В проекте представлен объектно-ориентированный подход в проектировании программного обеспечения информационного плана, дающий возможность на ранних этапах разработки учесть все нюансы будущей программы, необходимый набор функций, состав и структуру баз данных, что в дальнейшем исключает необходимость переработки уже написанных компонентов программы.

В программе присутствует простой и понятный пользовательский интерфейс, так что полностью разобраться во функционале программы не составит труда даже для неопытного пользователя.

Это приложение разработано для удобства тех, кто желает узнать больше марок и восполнить или начать собирать свою коллекцию. С помощью этой программы можно легко найти марки по многим критериям.

Планы на будущее: максимально расширить базу, создать расширенный поиск для более точного нахождения нужных пользователю марок, создать функцию «Избранное», куда пользователь мог бы заносить понравившиеся ему марки. Разработать аукцион марок.

ПЕРЕЧЕНЬ ССЫЛОК

1. Бондарев, В.М. Объектно-ориентированное программирование на C# [Текст]: учеб. пособ. /В.М. Бондарев. – Х.: Компания СМИТ, 2009 – 224 с.
2. Буч, Г. Объектно-ориентированный анализ и проектирование [Текст]: пер. с англ – М.: 000 "И.Д. Вильяме", 2008 - 720 с.
3. Мандер, Т. Разработка пользовательского интерфейса [Текст]: пер. с англ. – М.: ДМК Пресс, 2008 – 412 с.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ. – М.: Вильямс, 2013 – 1312 с.
5. Шилдт, Г. C# 4.0.: Полное руководство [Текст]: пер. с англ. – М.: Вильямс, 2011 – 1056 с.

Приложение А

Частичный код программы

Класс Collector

```
/// <summary>
    /// Класс, с помощью которого будет осуществляться работа с коллекционерами.
    /// </summary>
    [Serializable]
    class Collector
    {
        public string Country { set; get; }      // Страна
        public string Name { set; get; }         // Имя
        public string Phone { set; get; }        // телефон
        public int Id { set; get; }              // Номер коллекционера.
        public ListOfMarks Listmarks;            // Колекция марок

        /// <summary>
        /// Конструктор для создания нового коллекционера.
        /// </summary>
        /// <param name="n"></param>
        /// <param name="c"></param>
        /// <param name="p"></param>
        /// <param name="id"></param>
        public Collector(string n, string c, string p, int id)
        {
            Name = n;
            Phone = p;
            Country = c;
            Id = id;
            Listmarks = new ListOfMarks();
        }

        /// <summary>
        /// Конструктор по умолчанию.
        /// </summary>
        public Collector()
        {

        }
     }
```

Класс Marka

```
/// <summary>
    /// Класс, с помощью которого будет осуществляться работа с марками.
    /// </summary>
    [Serializable]
    class Marka

    {
        public string Country { set; get; }      // страна
        public string Nominal { set; get; }      // номинальная цена
        public string Year { set; get; }         // год выпуска
```

```
        public string Tirage { set; get; }        // тираж
        public string Special { set; get; }       // особенности
        public int Id { set; get; }               // Номар марки
        public Collector Coll { set; get; }       // коллекционер

        /// <summary>
        /// Конструктор.
        /// </summary>
        /// <param name="cou"></param>
        /// <param name="nomin"></param>
        /// <param name="yea"></param>
        /// <param name="tirag"></param>
        /// <param name="specia"></param>
        /// <param name="id"></param>
        /// <param name="c"></param>
        public Marka(string cou, string nomin, string yea, string tirag, string
specia, int id, Collector c)
        {
            Country = cou;
            Nominal = nomin;
            Year = yea;
            Tirage = tirag;
            Special = specia;
            Id = id;
            Coll = c;
        }

        /// <summary>
        /// Конструктор по умолчанию.
        /// </summary>
        public Marka()
        {

        }

        /// <summary>
        /// Перекрытие метода ToString().
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return String.Format(" Страна: {0}, Номинальная стоимость: {1}, Год:
{2}, Тираж: {3}, Особенность: {4} ", Country, Nominal, Year, Tirage, Special);
        }
    }
```

Класс ListOfCollectors

```
/// <summary>
    /// Класс, с помощью которого совершается управление списком коллекционеров.
    /// </summary>

    [Serializable]
    class ListOfCollectors : List<Collector>
    {
        /// <summary>
        /// Проверка на то, если ли уже данный коллекционер в списке.
        /// </summary>
        /// <param name="ic"></param>
```

```
        /// <returns></returns>

public bool IsCopy(Collector ic)
        {
            foreach (Collector a in Serial.collectorsList)
            {
                if (a.Phone == ic.Phone)
                {
                    return true;
                }
            }
            return false;
        }
    }
```

Класс ListOfMark

```
/// <summary>
    /// Класс, с помощью которого совершается управление списком марок.
    /// </summary>

    [Serializable]
    class ListOfMarks : List<Marka>
    {

    }
```

Класс Serial

```
/// <summary>
    /// Класс для сериализации списков марок и коллекционеров
    /// </summary>

    [Serializable]
    class Serial
    {

        public static ListOfMarks marksList = new ListOfMarks();
//общий список марок
        public static ListOfCollectors collectorsList = new ListOfCollectors();
//общий список коллекционеров


        /// <summary>
        /// Сохранение изменений в общем списке коллекционеров.
        /// </summary>
        public static void SaveCollector()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Collectors.dat",
FileMode.OpenOrCreate))
                {
                    binFormat.Serialize(fStream, collectorsList);
                }
            }
            catch { }
```

```
        }

        /// <summary>
        /// Сооздание или открытие общего списка коллекционеров.
        /// </summary>
        public static void OpenCollector()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Collectors.dat",
FileMode.Open))
                {
                    collectorsList =
(ListOfCollectors)binFormat.Deserialize(fStream);
                }
            }
            catch { }
        }

        /// <summary>
        /// Сооздание или открытие общего списка марок.
        /// </summary>
        public static void OpenMark()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Marks.dat",
FileMode.Open))
                {
                    marksList = (ListOfMarks)binFormat.Deserialize(fStream);
                }
            }
            catch { }
        }

        /// <summary>
        /// Сохранение изменений в общем списке марок.
        /// </summary>
        public static void SaveMark()
        {
            BinaryFormatter binFormat = new BinaryFormatter();
            try
            {
                using (Stream fStream = new FileStream("Marks.dat",
FileMode.OpenOrCreate))
                {
                    binFormat.Serialize(fStream, marksList);
                }
            }
            catch { }
        }
    }
```

Форма About

```
public partial class About : Form
    {
        public About()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Нажатие на кнопку "На главную", котороя закрывает данное окно
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void okButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
```

Форма AddCollector

```
[Serializable]
    public partial class AddCollector : Form
    {

        Collector temp;     // буферная переменная
        private int id;     // номер коллекционера в общем списке

        public AddCollector()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Запуск формы.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void AddCollector_Load(object sender, EventArgs e)
        {
            Serial.OpenCollector();
        }

        /// <summary>
        /// Кнопка "Добавить": добавление коллекционера.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void addButton_Click(object sender, EventArgs e)
        {
            if (!this.Check())
            {
                MessageBox.Show("Не все поля заполнены!");
```

```
            }
            else if (nameTextBox.Text.Length <= 1)
            {
                MessageBox.Show("Имя не может быть одной буквой.");
                return;
            }
            else
            {
                id = Serial.collectorsList.Count;
                temp = new Collector(nameTextBox.Text, countryTextBox.Text,
phoneTextBox.Text, id);
                if (Serial.collectorsList.IsCopy(temp))
                {
                    MessageBox.Show("Этот коллекционер уже существует.");
                    return;
                }
                else
                {
                    Serial.collectorsList.Add(temp);
                    MessageBox.Show("Клиент добавлен");
                    Serial.SaveCollector();
                }
            }
            this.ClearAll();
            this.Close();
        }

        /// <summary>
        /// Кнопка "Отмена".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void cancelButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        /// <summary>
        /// Проверка на то, заполнены ли поля.
        /// </summary>
        private bool Check()
        {
            if (String.IsNullOrWhiteSpace(nameTextBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(countryTextBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(phoneTextBox.Text)) return false;
            return true;
        }

        /// <summary>
        /// Очистка всех полей.
        /// </summary>
        private void ClearAll()
        {
            nameTextBox.Text = "";
            countryTextBox.Text = "";
            phoneTextBox.Text = "";
        }
```

```
        /// <summary>
        /// Проверка правильности ввода в графе "Страна".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void countryTextBox_TextChanged(object sender, EventArgs e)
        {
            if
(!System.Text.RegularExpressions.Regex.IsMatch(countryTextBox.Text, "^[a-zA-Za-
яА-Я]*$"))
            {
                countryTextBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только буквы.");
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Имя".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void nameTextBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(nameTextBox.Text,
"^[a-zA-Za-яА-Я]*$"))
            {
                nameTextBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только буквы.");
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Телефон".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void phoneTextBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(phoneTextBox.Text,
"^[0-9]*$"))
            {
                phoneTextBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }
        }
    }
```

Форма AddMark

```
[Serializable]
    public partial class AddMark : Form
    {
        Marka temp;          //буферная переменная
        private int id;      //номер марки в общем списке

        public AddMark()
        {
            InitializeComponent();
```

```
        }

        /// <summary>
        /// Запуск формы.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void AddMark_Load(object sender, EventArgs e)
        {
            Serial.OpenMark();
            Serial.OpenCollector();
            foreach (Collector name in Serial.collectorsList)
            {
                collectorBox.Items.Add(name.Name);
            }
        }

        /// <summary>
        /// Кнопка "Добавить марку".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void AddButton_Click(object sender, EventArgs e)
        {
            if (!Check())
            {
                MessageBox.Show("Не все поля заполнены!");
                return;
            }
            else if (!CheckYear())
            {
                MessageBox.Show("Введена неверная дата");
                return;
            }
            else
            {
                Serial.OpenCollector();
                Serial.OpenMark();
                id = Serial.marksList.Count;
                temp = new Marka(countryBox.Text, nominalBox.Text, yearBox.Text,
tirageBox.Text,
                        specialBox.Text, id,
Serial.collectorsList[collectorBox.SelectedIndex]);
                Serial.marksList.Add(temp);

Serial.collectorsList[Serial.collectorsList[collectorBox.SelectedIndex].Id].List
marks.Add(temp);
                MessageBox.Show("Марка добавлена");
            }
            Serial.SaveCollector();
            Serial.SaveMark();
            this.ClearAll();
            this.Close();
        }

        /// <summary>
        /// Кнопка "Отмена".
        /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        /// <summary>
        /// Очистка всех полей.
        /// </summary>
        private void ClearAll()
        {
            collectorBox.SelectedIndex = -1;
            countryBox.Text = "";
            yearBox.Text = "";
            nominalBox.Text = "";
            tirageBox.Text = "";
            specialBox.Text = "";
        }

        /// <summary>
        /// Проверка на то, заполнены ли поля.
        /// </summary>
        /// <returns></returns>
        private bool Check()
        {
            if (collectorBox.SelectedIndex == -1) return false;
            if (String.IsNullOrWhiteSpace(countryBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(yearBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(nominalBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(tirageBox.Text)) return false;
            if (String.IsNullOrWhiteSpace(specialBox.Text)) return false;
            return true;
        }

        /// <summary>
        /// Проверка на правильность вводимой даты.
        /// </summary>
        /// <returns></returns>
        private bool CheckYear()
        {
            if (Convert.ToInt32(yearBox.Text) > DateTime.Now.Year ||
Convert.ToInt32(yearBox.Text) < 1840)
            {
                yearBox.Text = string.Empty;
                return false;
            }
            else
                return true;
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Стоимость".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void nominalBox_TextChanged(object sender, EventArgs e)
        {
```

```
            if (!System.Text.RegularExpressions.Regex.IsMatch(nominalBox.Text,
"^[0-9]*$"))
            {
                nominalBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Страна".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void countryBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(countryBox.Text,
"^[a-zA-Za-яA-Я]*$"))
            {
                countryBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только буквы.");
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Тираж".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tirageBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(tirageBox.Text,
"^[0-9]*$"))
            {
                tirageBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Год".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void yearBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(yearBox.Text,
"^[0-9]*$"))
            {
                yearBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }

        }
    }
```

Форма AllCollectors

```
public partial class AllCollectors : Form
    {
        private int n;      //определение критерия поиска

        /// <summary>
        /// Открытие и заполнение таблицы.
        /// </summary>
        public AllCollectors()
        {
            InitializeComponent();
            searchBox.Items.Add("По стране");
            searchBox.Items.Add("По имени");
            searchBox.Items.Add("По телефону");
            FillGridView();
        }

        /// <summary>
        /// Заполнение данными таблицу и добавление категории поиска.
        /// </summary>
        public void FillGridView()
        {
            collectorGridView.Rows.Clear();
            Serial.OpenCollector();
            foreach (Collector collector in Serial.collectorsList)
            {
                collectorGridView.Rows.Add(collector.Id, collector.Name,
collector.Country, collector.Phone, collector.Listmarks.Count);
            }

        }

        /// <summary>
        /// Кнопка "На главную".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void main_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        /// <summary>
        /// Редактирование телефона коллекционера.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void collectorGridView_CellEndEdit(object sender,
DataGridViewCellEventArgs e)
        {

            DataGridViewRow selectedRow =
collectorGridView.Rows[collectorGridView.SelectedCells[0].RowIndex];
            string a = Convert.ToString(selectedRow.Cells["phone"].Value);
            try
            {
                Convert.ToInt64(a);
            }
            catch (FormatException)
```

```
            {
                MessageBox.Show("В телефоне могут быть только числа.");
            }

            foreach (Collector coll in Serial.collectorsList)
            {
                if (coll.Phone == a)
                {
                    MessageBox.Show("Этот коллекционер уже существует.");
                    selectedRow.Cells["phone"].Value = "";
                }
                else
                {
                    continue;
                }
            }

            Serial.collectorsList.Find(coll => coll.Id ==
Convert.ToInt32(selectedRow.Cells["id"].Value)).Phone = a;
        }

        /// <summary>
        /// Кнопка "Сохранить изменения".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void savebutton_Click(object sender, EventArgs e)
        {
            Serial.SaveCollector();
        }

        /// <summary>
        /// Определение критерия поиска.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void searchBox_TextChanged(object sender, EventArgs e)
        {
            if (searchBox.Text == "По стране") n = 1;
            if (searchBox.Text == "По имени") n = 2;
            if (searchBox.Text == "По телефону") n = 3;
            searchtextBox.Visible = true;
        }

        /// <summary>
        /// Заполнение данными таблицу после поиска.
        /// </summary>
        /// /// <param name="a"></param>
        private void FillGridViewSearch(IEnumerable<Collector> a)
        {
            collectorGridView.Rows.Clear();
            foreach (Collector coll in a)
            {
                collectorGridView.Rows.Add(coll.Id, coll.Name, coll.Country,
coll.Phone, coll.Listmarks.Count);
            }
        }
```

```
        /// <summary>
        /// Кнопка "Поиск".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void searchButton_Click(object sender, EventArgs e)
        {
            FillGridViewSearch(SearchText());
        }

        /// <summary>
        /// Поиск.
        /// </summary>
        private ListOfCollectors SearchText()
        {
            ListOfCollectors searchList = new ListOfCollectors();
            if (searchtextBox.Text.Length == 0)
            {
                MessageBox.Show("Вы ничего не ввели");
            }
            else
            {
                foreach (Collector coll in Serial.collectorsList)
                {
                    if ((n == 1 &&
coll.Country.ToLower().Contains(searchtextBox.Text.ToLower())) || (n == 2 &&
coll.Name.ToLower().Contains(searchtextBox.Text.ToLower())) || (n == 3 &&
coll.Phone.ToString().ToLower().Contains(searchtextBox.Text.ToLower())))
                    {
                        searchList.Add(coll);
                    }
                    else
                    {
                        continue;
                    }
                }
            }
            return searchList;
        }

        /// <summary>
        /// Кнопка "Удалить коллекционера".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void deleteButton_Click(object sender, EventArgs e)
        {
            Serial.OpenCollector();
            Serial.OpenMark();
            DataGridViewRow selectedRow =
collectorGridView.Rows[collectorGridView.SelectedCells[0].RowIndex];

            foreach (Marka mark in Serial.collectorsList.Find(coll => coll.Id ==
Convert.ToInt32(selectedRow.Cells["id"].Value)).Listmarks)
            {
                Serial.marksList.Remove(Serial.marksList.Find(marka =>
marka.Coll.Name == Serial.collectorsList.Find(coll => coll.Id ==
Convert.ToInt32(selectedRow.Cells["id"].Value)).Name));
```

```
            }

            Serial.collectorsList.Find(coll => coll.Id ==
Convert.ToInt32(selectedRow.Cells["id"].Value)).Listmarks.Clear();
            Serial.collectorsList.Remove(Serial.collectorsList.Find(coll =>
coll.Id == Convert.ToInt32(selectedRow.Cells["id"].Value)));

            Serial.SaveCollector();
            Serial.SaveMark();
            FillGridView();
        }

        /// <summary>
        /// Появление кнопки "Удалить коллекционера" после нажатия на ячейку
коллекционера.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void collectorGridView_Click(object sender, EventArgs e)
        {
            deleteButton.Visible = true;
        }

        /// <summary>
        /// Кнопка "Показать все".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button2_Click(object sender, EventArgs e)
        {
            FillGridView();
        }
    }
```

Форма AllMark

```
public partial class AllMark : Form
    {
        private int n;            //определение критерия поиска

        /// <summary>
        /// Открытие и заполнение таблицы.
        /// </summary>
        public AllMark()
        {
            InitializeComponent();
            searchBox.Items.Add("По году");
            searchBox.Items.Add("По стоимости");
            searchBox.Items.Add("По тиражу");
            searchBox.Items.Add("По коллекционеру");
            searchBox.Items.Add("По теме");
            searchBox.Items.Add("По стране");
            FillGridView();
        }

        /// <summary>
        /// Заполнение данными таблицу и добавление категории поиска.
```

```
        /// </summary>
        public void FillGridView()
        {
            MarksGridView.Rows.Clear();
            Serial.OpenMark();
            foreach (Marka marka in Serial.marksList)
            {
                MarksGridView.Rows.Add(marka.Id, marka.Country, marka.Year,
marka.Nominal, marka.Tirage, marka.Special, marka.Coll.Name);
            }
        }

        /// <summary>
        /// Кнопка "На главную".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void main_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        /// <summary>
        /// Определение критерия поиска.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void searchBox_TextChanged(object sender, EventArgs e)
        {
            if (searchBox.Text == "По году") n = 1;
            if (searchBox.Text == "По стоимости") n = 2;
            if (searchBox.Text == "По тиражу") n = 3;
            if (searchBox.Text == "По коллекционеру") n = 4;
            if (searchBox.Text == "По теме") n = 5;
            if (searchBox.Text == "По стране") n = 6;
        }

        /// <summary>
        /// Открытие и скрытие полей по типу критерия поиска.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void searchBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (n)
            {
                case 1:
              case 2:
              case 3:
                  VisDigits();
                  textbox.Visible = false;
                  break;
              case 4:
              case 5:
              case 6:
                  NonVisDigits();
                  textbox.Visible = true;
                  break;
```

```
            }
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Минимальное значение".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void mintextBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(mintextBox.Text,
"^[0-9]*$"))
            {
                mintextBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }
        }

        /// <summary>
        /// Открытие полей для поиска числовых значений.
        /// </summary>
        private void VisDigits()
        {
            label1.Visible = true;
            label2.Visible = true;
            mintextBox.Visible = true;
            maxtextBox.Visible = true;
        }

        /// <summary>
        /// Скрытие полей для поиска числовых значений.
        /// </summary>
        private void NonVisDigits()
        {
            label1.Visible = false;
            label2.Visible = false;
            mintextBox.Visible = false;
            maxtextBox.Visible = false;
        }

        /// <summary>
        /// Проверка правильности ввода в графе "Максимальное значение".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void maxtextBox_TextChanged(object sender, EventArgs e)
        {
            if (!System.Text.RegularExpressions.Regex.IsMatch(maxtextBox.Text,
"^[0-9]*$"))
            {
                maxtextBox.Text = string.Empty;
                MessageBox.Show("Можно вводить только цифры.");
            }
        }

        /// <summary>
        /// Заполнение данными таблицу после поиска.
        /// </summary>
```

```
        /// /// <param name="a"></param>
        private void FillGridViewSearch(IEnumerable<Marka> a)
        {
            MarksGridView.Rows.Clear();
            foreach (Marka marka in a)
            {
                MarksGridView.Rows.Add(marka.Id, marka.Country, marka.Year,
marka.Nominal, marka.Tirage, marka.Special, marka.Coll.Name);
            }
        }

        /// <summary>
        /// Кнопка "Найти".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button1_Click(object sender, EventArgs e)
        {
            if (n == 4 || n == 5 || n == 6)
            {
                FillGridViewSearch(SearchText());
            }
            else
            {
                FillGridViewSearch(SearchDigits());
            }

        }

        /// <summary>
        /// Поиск по текстовым значениям.
        /// </summary>
        private ListOfMarks SearchText()
        {
            ListOfMarks searchList = new ListOfMarks();
            if (textbox.Text.Length == 0)
            {
                MessageBox.Show("Вы ничего не ввели");
            }
            else
            {
                foreach (Marka mark in Serial.marksList)
                {
                    if ((n == 4 &&
mark.Coll.Name.ToLower().Contains(textbox.Text.ToLower())) || (n == 5 &&
mark.Special.ToLower().Contains(textbox.Text.ToLower())) || (n == 6 &&
mark.Country.ToLower().Contains(textbox.Text.ToLower())))
                    {
                        searchList.Add(mark);
                    }
                    else
                    {
                        continue;
                    }
                }
            }
            return searchList;
        }
```

```
        /// <summary>
        /// Поиск по цифровым значениям.
        /// </summary>
        private IEnumerable<Marka> SearchDigits()
        {

            if (mintextBox.Text.Length == 0 || maxtextBox.Text.Length == 0)
            {
                MessageBox.Show("Вы ничего не ввели");
                return null;
            }
            else
            {
                switch (n)
                {
                    case 1:
                        var searchList = Serial.marksList.Where(mark =>
Convert.ToInt32(mark.Year) >= Convert.ToInt32(mintextBox.Text) &&
Convert.ToInt32(mark.Year) <= Convert.ToInt32(maxtextBox.Text));
                        return searchList;
                    case 2:
                        searchList = Serial.marksList.Where(mark =>
Convert.ToInt32(mark.Nominal) >= Convert.ToInt32(mintextBox.Text) &&
Convert.ToInt32(mark.Nominal) <= Convert.ToInt32(maxtextBox.Text));
                        return searchList;
                    case 3:
                        searchList = Serial.marksList.Where(mark =>
Convert.ToInt32(mark.Tirage) >= Convert.ToInt32(mintextBox.Text) &&
Convert.ToInt32(mark.Tirage) <= Convert.ToInt32(maxtextBox.Text));
                        return searchList;
                    default:
                        return null;
                }
            }
        }
        /// <summary>
        /// Кнопка "Показать все".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button2_Click(object sender, EventArgs e)
        {
            FillGridView();
        }
```

Форма Form1

```
using System.IO;

namespace Filatelist1
{
    [Serializable]
    public partial class Form1 : Form
    {

        public Form1()
        {
```

```
            InitializeComponent();
        }

        /// <summary>
        /// Кнопка меню "Добавить..." -> "Коллекционера". Открывает окно
добавления коллекционера.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void коллекционераToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AddCollector AC = new AddCollector();
            AC.Show();
        }

        /// <summary>
        /// Кнопка меню "Добавить..." -> "Марку". Открывает окно добавления
марки.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void маркуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AddMark AM = new AddMark();
            AM.Show();
        }

        /// <summary>
        /// Кнопка меню "Выход".Закрывает программу.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            RefreshForm();
        }

        /// <summary>
        /// Обновление списков марок и коллекционеров.
        /// </summary>
        private void RefreshAll()
        {
            Serial.OpenMark();
            Serial.OpenCollector();
        }

        /// <summary>
        /// Обновление/заполнение списка коллекционеров в главном меню.
        /// </summary>
        public void RefreshForm()
        {
            RefreshAll();
```

```
            foreach (Collector name in Serial.collectorsList)
            {
                listBox1.Items.Add(name.Name);
            }
            listBox2.Items.Clear();
        }

        /// <summary>
        /// Появление списка марок в listBox2, при выборе коллекционера в
listBox1.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

            RefreshAll();
            listBox2.Items.Clear();
            ClearLabel();

            try
            {

                if
(Serial.collectorsList[listBox1.SelectedIndex].Listmarks.Count == 0)
                {
                    MessageBox.Show("У данного коллекционера еще нет марок");
                }
                else
                {
                    int sum = 0;

                    foreach (Marka mark in
Serial.collectorsList[listBox1.SelectedIndex].Listmarks)
                    {
                        listBox2.Items.Add(mark.Special);
                        sum += Convert.ToInt32(mark.Nominal);
                    }
                    priceLabel.Text = sum.ToString();
                }
            }
            catch (ArgumentOutOfRangeException)
            {

            }
        }

        /// <summary>
        /// Обновление списка коллекционеров в главном меню.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            RefreshForm();
        }
```

```
        /// <summary>
        /// Кнопка меню "Марки". Открывает окно общего списка марок.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void маркиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AllMark AM = new AllMark();
            AM.Show();
        }

        /// <summary>
        /// Кнопка меню "Коллекционеры". Открывает окно общего списка
коллекционеров.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void коллекционерыToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AllCollectors AC = new AllCollectors();
            AC.Show();
        }

        /// <summary>
        /// Появление информации о марке, при выборе марки в listBox2.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            RefreshAll();
            try
            {
                yearLabel.Text =
Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s[Serial.marksList[listBox2.SelectedIndex].Id].Year;
                nominalLabel.Text =
Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s[Serial.marksList[listBox2.SelectedIndex].Id].Nominal;
                countryLabel.Text =
Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s[Serial.marksList[listBox2.SelectedIndex].Id].Country;
                tirageLabel.Text =
Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s[Serial.marksList[listBox2.SelectedIndex].Id].Tirage;
                specialLabel.Text =
Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s[Serial.marksList[listBox2.SelectedIndex].Id].Special;
                deleteButton.Visible = true;
            }
            catch (ArgumentOutOfRangeException)
            {
                ClearLabel();
            }
        }

        /// <summary>
```

```
        /// Кнопка меню "О программе...". Открывает окно информации про
программу.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            About Ab = new About();
            Ab.Show();
        }

        /// <summary>
        /// Очистка полей.
        /// </summary>
        private void ClearLabel()
        {
            yearLabel.Text = "";
            nominalLabel.Text = "";
            countryLabel.Text = "";
            tirageLabel.Text = "";
            specialLabel.Text = "";
            deleteButton.Visible = false;
            priceLabel.Text = "";
        }

        /// <summary>
        /// Кнопка "Удалить марку из коллекции".
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button2_Click(object sender, EventArgs e)
        {
            Serial.OpenCollector();
            Serial.OpenMark();

Serial.collectorsList[Serial.collectorsList[listBox1.SelectedIndex].Id].Listmark
s.Remove(Serial.collectorsList[listBox1.SelectedIndex].Listmarks[Serial.marksLis
t[listBox2.SelectedIndex].Id]);
            Serial.marksList[listBox2.SelectedIndex].Coll.Name = "Сохранено в
базе";
            Serial.SaveMark();
            Serial.SaveCollector();
        }

        /// <summary>
        /// Кнопка меню "Сохранить в TXT". Сохраняет базу марок в локальном
диске D.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void сохранитьВTXTToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using (StreamWriter writer = new StreamWriter("d:\\ListOfMarks.txt",
false, Encoding.UTF8))
            {
                foreach (Marka mark in Serial.marksList)
```

```
                {
                    writer.WriteLine(mark.ToString());
                }
            }
            MessageBox.Show("Сохранение прошло успешно.");
        }

        /// <summary>
        /// Кнопка меню "Помощь". Открывает окно помощи программы.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            HelpForm HF = new HelpForm();
            HF.Show();

        }
    }
}
```

Форма HelpForm

```
public partial class HelpForm : Form
    {
        public HelpForm()
        {
            InitializeComponent();
        }
    }
```

Класс Program

```
class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }

    }
```